Discrete Applied Mathematics 16 (1987) 17–30
North-Holland

STRING GRAMMARS WITH DISCONNECTING OR A BASIC ROOT OF THE DIFFICULTY IN GRAPH GRAMMAR PARSING

Klaus-Jörn LANGE
Fachbereich Informatik der Universität Hamburg, Rothenbaumchaussee 67–69, 2000 Hamburg 13, Fed. Rep. of Germany

Emo WELZL*
Institute of Applied Mathematics and Computer Science, University of Leiden, The Netherlands

Received 19 March 1985
Revised 12 February 1986

The complexity of languages generated by so-called context-free string grammars with disconnecting is investigated. The result is then applied to a number of graph grammar models with finite Church Rosser property. In particular, it is shown that these graph grammars can generate NP-complete languages.

Introduction

Graph grammars are commonly considered as rewriting systems which generate languages with a 'hard' membership problem. This is, for example, supported by the fact that node label controlled graph grammars, NLC grammars for short, can generate PSPACE-complete graph languages (see Janssens & Rozenberg, 1980b; Brandenburg, 1983). Moreover, it was shown in Brandenburg (1983) and Túran (1983) that NLC grammars without 'chain-rules' can generate NP-complete languages. On the one hand, these results are surprising, since NLC grammars are 'context-free' in the sense that a production of an NLC grammar has a (single) label as left hand side and such a production is applicable to a node in a graph if and only if this node is labelled by the label representing the left hand side of the production. That is, there is no context-sensitive condition for the applicability of a production to a node. On the other hand, 'context-sensitivity' is hidden in the embedding mechanism of NLC grammars: Instead of checking the presence of certain labels in the neighbourhood of the node to be replaced, the embedding mechanism disconnects to all 'undesired' labels. A consequence of this 'context-sensitivity' via the embedding mechanism is that NLC grammars do not satisfy the finite Church Rosser property, from now on briefly fCR. (Roughly speaking, the

* On leave from: Institutes for Information Processing, IIG, Technical University of Graz and Austrian Computer Society, Schießstattgasse 4a, A-8010 Graz, Austria.

fCR means that "nonoverlapping rewriting steps can be done in any order": see Brandenburg, 1983, or more formally in Rozenberg and Welzl, 1984, Lemma 2.3).

Although there are already a number of investigations of the syntax analysis problem for graph grammars (see, e.g., Della Vigna & Ghezzi, 1978; Frank, 1978; Silisenko, 1982; Kaul, 1983; Brandenburg, 1982, 1983; Turán, 1983), there is no proof so far that graph grammars with fCR can generate hard languages, e.g., NP-hard languages. So, for example, Brandenburg (1983) supposed that "... in an appropriate framework of graph grammars the existence of the fCR guarantees a polynomial membership, ...".

On the other side there was at least some indication in Rozenberg & Welzl (1984), that the fCR alone might not be sufficient to guarantee polynomial time membership: it was shown that so-called 'unlabelled' boundary NLC languages have a polynomial time membership problem for connected graphs of (fixed) bounded degree, while the membership problem can be NP-complete for (possibly) disconnected graphs of (fixed) bounded degree and for connected graphs of arbitrary degree. (Boundary NLC grammars are a subclass of NLC grammars which have the fCR. Unlabelled boundary NLC languages are the corresponding languages where the labels in the graphs are omitted.)

Thus two additional properties appear to be important for the membership complexity: connectedness and maximal degree of the considered graphs. The goal of this paper is to investigate this in detail, i.e., to show that these properties are really crucial for membership problems of languages generated by graph grammars with fCR; in particular, we concentrate on connectedness.

The basic idea is as follows: we try to go a simple step from 'conventional' context-free string grammars towards graph grammars by supplying these string grammars with the power of disconnecting. This is illustrated as follows:

Let ¢ be a special symbol which means 'disconnect' or 'cut'. Consider now a production of the form $A \to ¢aA¢ab¢aa$ and apply this production to the string $aabAb$. Then this results in the multiset $[aab, aab, ab, aA]$, rather than in the string $aab¢aA¢ab¢aab$, because we disconnect where ¢ occurs. Obviously, in such a grammar with disconnecting also the left-hand side of a derivation step will be a multiset. So we will typically have derivation steps like $[aabAb, ab] \Rightarrow [aab, aab, ab, ab, aA]$. A similar idea can be found in Ruohonen (1975a,b). (See also Rozenberg & Salomaa 1980, p. 78.)

We will demonstrate that:

(i) There is a linear context-free string grammar with disconnecting which generates an NP-complete language, while (ii) the languages of right-linear string grammars with disconnecting are still recognizable in linear time.

We will show that the result (i) can be applied to a number of graph grammars with fCR:

(1) There is a context-free edge-replacement grammar (as defined in Kreowski, 1979) generating an NP-complete language of graphs with maximal degree at most 2.

(2) There is a context-free node-replacement grammar (as defined in Pratt, 1971) generating an NP-complete language of graphs with maximal degree at most 2.

(3a) There is a boundary NLC grammar (as defined in Rozenberg & Welzl, 1984) generating an NP-complete language of graphs with maximal degree at most 2.

(3b) There is a boundary NLC grammar generating an NP-complete language of connected graphs.

In (1), (2), and (3a) it is necessary that the considered graph languages may contain disconnected graphs. So our approach does not apply to context-free node-replacement grammars as defined by Della Vigna & Ghezzi (1978), because they require the right-hand side of a production to be a connected graph (otherwise their definition coincides with Pratt's). Moreover, boundary NLC languages have a polynomial time membership for connected graphs of (fixed) bounded degree, also in the labelled case (Rozenberg & Welzl, 1984).

Possibly, similar results might hold or the models in (1) and (2) if one applies a dynamic programming approach to the membership problem.

The paper is organized as follows: In Section 1 we consider string grammars with disconnecting. Actually, we will treat normal string grammars which generate languages over an alphabet which contains the cut symbol ¢. Then we 'cut' a word w of the language where ¢ occurs to obtain the above mentioned multiset, called the cut image of w. It is obvious that the postponement of the cutting to the final product of a derivation is only a technical difference. We will show that there is a linear (deterministic) context-free language K the cut image of which is NP-complete. Moreover, we will demonstrate that the cut image of a regular language has a linear time membership problem. In Section 2 will briefly indicate the applications of our result to some graph grammar models as mentioned in points (1), (2), (3a), and (3b) above. Finally, in Section 3, we conclude with a short discussion.

We assume the reader to be familiar with basic formal language theory (e.g., in the scope of Salomaa, 1973, or Bucher & Maurer, 1984) and with basics about NP-completeness (see Garey & Johnson, 1979).

1. String grammars with disconnecting

In this section we first show that context-free string grammars can generate NP-complete languages, if they are additionally equipped with the 'power of disconnecting'. Actually, we will show that there is even a linear and deterministic context-free string grammar with disconnecting which generates an NP-complete language. Further on, we demonstrate that languages of regular string grammars with disconnecting are still recognizable in linear time.

As indicated in the introduction, for the sake of simplicity we will postpone the disconnecting mechanism during a derivation until we have the final result of the derivation. This will be done by inserting a special *cut symbol* ¢ during a derivation, where we want to disconnect. Then we cut the derived terminal word into pieces by

building the (finite) multiset of all maximal subwords not containing the cut symbol ¢.

Finite multisets over a set M can be regarded either as mappings from M to $\mathbb{N}_0$ with a finite support or as equivalence classes over finite sequences over M (where two sequences are equivalent if one is a reordering of the other). Both possibilities are equivalent but lead to different encodings. In the first case a multiset A would be described by a finite list of words together with their corresponding degrees of membership in A, while in the second case any (for instance lexicographically ordered) sequence, contained in the equivalence class A, could represent A. In the following we use the second possibility since both the reduction from the complexity-theoretical side and the reduction to the graph-theoretical side favour this approach in a natural way.

Now the formal process of disconnecting will be done as follows: Throughout this section let ¢ be a distinguished symbol. Let now Σ be an alphabet and let w be a word over $\Sigma \cup \{¢\}$. (Whenever we write $\Sigma \cup \{¢\}$ we assume $¢ \notin \Sigma$.) Then w has a unique decomposition of the form $w = w_1 ¢ w_2 ¢ \cdots ¢ w_n$, $n \geq 1$, where $w_i \in \Sigma^*$ for all i. The *cut image* of w, cut(w) for short, is then the multiset consisting of all w_i, $1 \leq i \geq n$, with $w_i \neq \lambda$. Take for example $w = 01¢¢10¢01¢000$, then cut(w) = [000, 01, 01, 10]. (We use the brackets '[' and ']' to indicate a multiset). For a language L over $\Sigma \cup \{¢\}$, its *cut image*, CUT(L) for short, is the set $\{\text{cut}(w) \mid w \in L\}$. It is easy to show that CUT(L) is in NP, if L is a context-free language. The first goal of this section is now to define a linear context-free language K such that CUT(K) is NP-complete. This will be done via a reduction of the Hamiltonian path problem to the membership problem for CUT(K). To this end, we need the following notion. Let u and w be words over 0 and 1. We say that u and w are *word-adjacent*, if they are of the same length and if they have a 1 in the same position, i.e., if there exist $i, j \geq 0$ such that u and w are in $\{0,1\}^i 1 \{0,1\}^j$.

The crucial language K is now a language over $\{a, b, 0, 1, c\}$ defined as follows:

$$K := \{a^{i_1} w_1 ¢ a^{i_2} w_2 ¢ a^{i_3} \cdots w_{n-1} ¢ a^{i_n} w_n ¢ b^{i_n} u_n ¢ \cdots b^{i_2} u_2 ¢ b^{i_1} u_1 \mid$$
$$n \geq 1,\ i_j \geq 1 \text{ and } w_j, u_j \in \{0,1\}^+ \text{ for all } 1 \leq j \leq n \text{ and } w_j \text{ is word-adjacent to the mirror image of } u_{j+1} \text{ for all } j,\ 1 \leq j \leq n-1\}.$$

Lemma 1.1. *K is a linear context-free language.*

Proof. Let $G_0 = (N_0, \Sigma_0, P_0, S_0)$ be a linear context-free grammar which is defined as follows. The set of nonterminals is $N_0 = \{S_0, A, B, C, D\}$. $\Sigma_0 = \{a, b, 0, 1, ¢\}$ is the set of terminals, S_0 is the start symbol and P_0 consists of the following productions:

$S_0 \to S_0 0 \mid S_0 1 \mid C0 \mid C1;$

$C \to aCb \mid aA\¢b \mid aD\¢b;$

$A \to 0A1 \mid 1A0 \mid 1B1 \mid 1\¢C1;$

$B \to iBj \mid i\¢Cj, \quad i,j \in \{0,1\};$

$D \to 0D \mid 1D \mid 0 \mid 1;$

Note that (i) starting from C, one generates words of the form $a^k A\¢b^k$ (or $a^k D\¢b^k$), $k \geq 1$, and (ii) starting from A, one generates words of the form $w\¢Cu$, where w and the mirror image of u are word-adjacent. Hence, together with Fig. 1, it is not difficult to see that G_0 generates the language K. Obviously, G_0 is a linear grammar, which implies the result. □

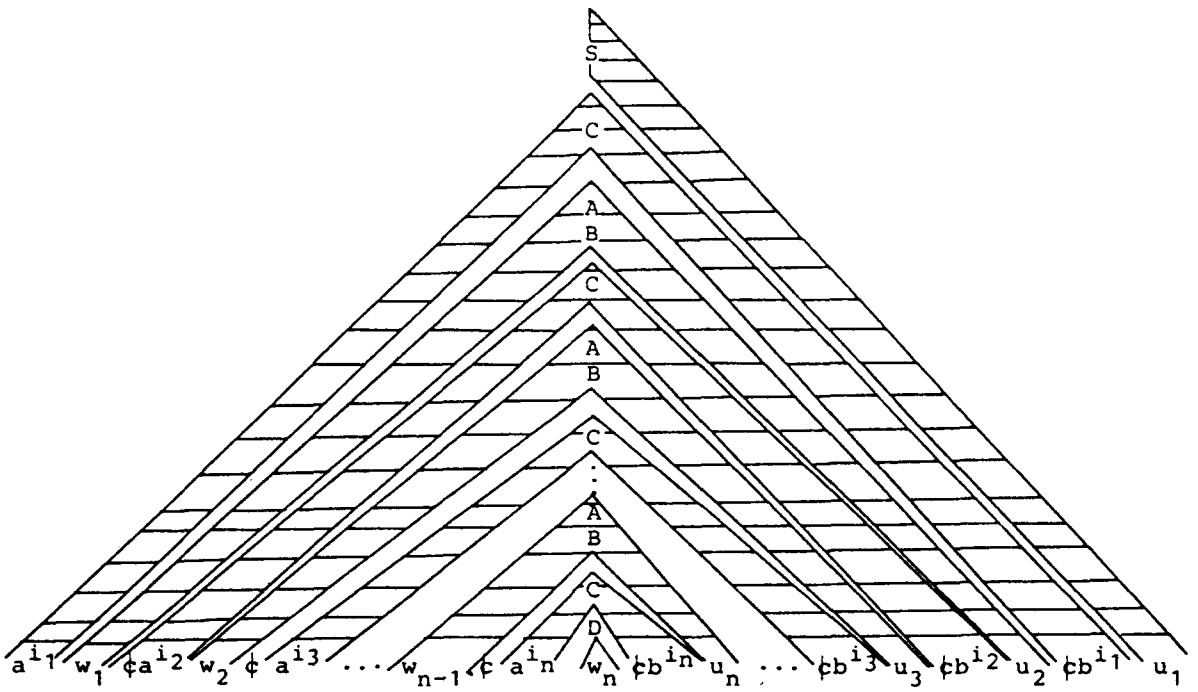


Fig. 1.

Remark. It is easily seen that K is also deterministic context-free!

In a next step we introduce now a representation of finite undirected unlabelled graphs as multisets of strings which has the following propety: (i) the representation is polynomial time computable from any standard representation of graphs and (ii) a graph has a Hamiltonian path if and only if its representation is in CUT(K).

Since the Hamiltonian path problem is NP-complete (see Garey & Johnson, 1979) this shows that CUT(K) is NP-complete.

Let $X=(V,E)$ be a graph, where V is a finite set and E is a set of two-element subsets of V. Let $V=\{v_1, v_2, \ldots, v_n\}$ and $E=\{e_1, \ldots, e_m\}$ be arbitrary but fixed

enumerations of the elements in V and E. For a node $v_i \in V$ set $h(v_i) = k_1 k_2 \cdots k_m$, where $k_j = 1$ if v_i and e_j are incident (i.e., $v_i \in e_j$), and $k_j = 0$, otherwise. In the following $\overleftarrow{h}(v_i)$ denotes the mirror image of $h(v_i)$. The *set representation* of X is the multiset

$$\mu(X) = [ah(v_1), aah(v_2), \ldots, a^n h(v_n), b\overleftarrow{h}(v_1), bb\overleftarrow{h}(v_2), \ldots, b^n \overleftarrow{h}(v_n)].$$

The connection between the representation $\mu(X)$ and the language K is given by the following crucial observation: two nodes v_i and v_j are adjacent in X if and only if $h(v_i)$ and $h(v_j)$ are word-adjacent.

Note that the set representation of X depends on the enumerations of V and E. Nevertheless, we refer to this representation by $\mu(X)$ (where we actually mean, that $\mu(X)$ is one of the possible set representations). Moreover, observe that $\mu(X)$ contains no element twice. But we defined it as a multiset in order to treat $\mu(X)$ as a possible element of CUT(K). Clearly, $\mu(X)$ can be computed in polynomial time from a graph X. It remains to show that for a graph X, $\mu(X)$ is in CUT(K) if and only if X contains a Hamiltonian path. This will be deone in the following two lemmata.

Lemma 1.2. *If $\mu(X) \in \mathrm{CUT}(K)$ for a graph X, then X contains a Hamiltonian path.*

Proof. Let $\mu(X) \in \mathrm{CUT}(K)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the enumeration of the node set of X underlying the representation $\mu(X)$. If $\mu(X) \in \mathrm{CUT}(K)$, then there is a permutation $(i_1, i_2, \ldots, i_n)$ of $(1, 2, \ldots, n)$ such that

$$a^{i_1} h(v_{i_1}) \not\in a^{i_2} h(v_{i_2}) \not\in \cdots \not\in a^{i_n} h(v_{i_n}) \not\in b^{i_n} \overleftarrow{h}(v_{i_n}) \not\in \cdots \not\in b^{i_1} \overleftarrow{h}(v_{i_1}) \in K.$$

This again holds only if $h(v_{i_j})$ is word-adjacent to the mirror image of $\overleftarrow{h}(v_{i_{j+1}})$, $1 \le j \le n-1$, which implies that v_{i_j} is adjacent to $v_{i_{j+1}}$. Consequently, $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ is a Hamiltonian path in X. □

Lemma 1.3. *If a graph X contains a Hamiltonian path, then $\mu(X) \in \mathrm{CUT}(K)$.*

Proof. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the enumeration of the node set of X underlying the representation $\mu(X)$ and let $(i_1, i_2, \ldots, i_n)$ be a permutation of $(1, 2, \ldots, n)$ such that $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ is a Hamiltonian path in X, i.e., v_{i_j} is adjacent to $v_{i_{j+1}}$, for all j, $1 \le j \le n-1$. Consider now the word

$$w = a^{i_1} h(v_{i_1}) \not\in a^{i_2} h(v_{i_2}) \not\in \cdots a^{i_n} h(v_{i_n}) \not\in b^{i_n} \overleftarrow{h}(v_{i_n}) \not\in \cdots \not\in b^{i_1} \overleftarrow{h}(v_{i_1}).$$

Then $h(v_{i_j})$ is word-adjacent to the mirror image of $\overleftarrow{h}(v_{i_{j+1}})$, for all j, $1 \le j \le n-1$. This shows $w \in K$ and hence $\mu(X) = \mathrm{cut}(w) \in \mathrm{CUT}(K)$. □

Now the following theorem can be directly obtained from Lemmata 1.1 through 1.3:

Theorem 1.4. *There exists a linear (deterministic) context-free string language K such that the membership problem for* CUT(K) *is NP-complete.*

In the next section we will see that this theorem entails NP-completeness results for a number of graph grammar models with fCR.

The natural follow-up question after Theorem 1.4 is "What is the complexity of the cut images of regular languages?". This will be treated in the remainder of this section.

We start with a simple lemma, the proof of which is obvious.

Lemma 1.5. *Let R be a regular language over $\Sigma \cup \{\cent\}$. Then there is a regular language $R' \subset (\cent\Sigma^+)^*$ with* $\mathrm{CUT}(R) = \mathrm{CUT}(R')$. □

For the rest of this section let R be an arbitrary but fixed regular language over $\Sigma \cup \{\cent\}$ and let $A = (Q, \Sigma \cup \{\cent\}, \delta, q_0, F\}$ be a finite complete deterministic automaton which accepts R. (Q is the set of states, $\Sigma \cup \{\cent\}$ the input alphabet, $\delta : Q \times \Sigma \cup \{\cent\} \to Q$ the transition function which is extended in the usual way to a function from $Q \times (\Sigma \cup \{\cent\})^*$ into Q, q_0 is the initial state and F is the set of accepting states). Let $Q = \{q_0, q_1, \ldots, q_{k-1}\}$ be a fixed enumeration of the states of Q, beginning with the initial state. Let w be a word in $\cent\Sigma^+$. Then let $f(w)$ be the vector

$$f(w) = (\delta(q_0, w), \delta(q_1, w), \ldots, \delta(q_{k-1}, w)),$$

i.e., f is a function from $\cent\Sigma^+$ into $\Gamma = Q^k$. Let w be a word in $(\cent\Sigma^+)^*$ and let $w = \cent w_1 \cent w_2 \cdots \cent w_n$, $n \geq 0$ be its unique decomposition with $w_i \in \Sigma^+$, for all i, $1 \leq i \leq n$. Then define $F(w)$ to be the word

$$F(w) = f(\cent w_1) f(\cent w_2) \cdots f(\cent w_n)$$

over Σ^*. Moreover, set $F(R) = \{F(w) \mid w \in R\}$. It is now easily seen that if for two words $w, w' \in (\cent\Sigma^+)^*$, $F(w) = F(w')$ holds, then $w \in R$ if and only if $w' \in R$. With that we can prove the following lemma.

Lemma 1.6. *Let $w \in (\cent\Sigma^+)^*$. Then* $\mathrm{cut}(w) \in \mathrm{CUT}(R)$ *if and only if $\psi(F(w)) \in \psi(F(R))$, where $\psi(F(w))$, $\psi(F(R))$ is the Parikh image of the word $F(w)$, of the language $F(R)$, respectively.*

Proof. *'if'* If $\psi(F(w)) \in \psi(F(R))$, then there is a $w' \in R$ such that $\psi(F(w)) = \psi(F(w'))$. This implies $w' \in (\cent\Sigma^+)^n$. If the unique decomposition of w' is $w' = \cent w_1' \cent w_2' \cdots \cent w_n'$, then there must be a permutation $(i_1, i_2, \ldots, i_n)$ of $(1, 2, \ldots, n)$ such that $f(\cent w_j) = f(\cent w_{i_j}')$. (Here $w = \cent c_1 \cent w_2 \cdots \cent w_n$ is the unique decomposition of w.) Now set $w'' = \cent w_{i_1} \cent w_{i_2} \cdots \cent w_{i_n}$. Then $F(w') = F(w'')$ and hence $w'' \in R$. But this implies $\mathrm{cut}(w) = \mathrm{cut}(w'') \in \mathrm{CUT}(R)$.

'only if' This is obvious. □

Now we can state the following theorem.

Theorem 1.7. *Membership in* CUT(R), *where* R *is a fixed regular language, can be decided in linear time, if the considered multiset* $M=[w_1, w_2, \ldots, w_n]$ *is given as a word* $w(M)=\¢w_1\¢w_2\cdots\¢w_n$.

Proof. Obviously, for $w\in(\¢\Sigma^+)^*$, $\psi(F(w))$ can be computed in linear time. Moreover, $F(R)$ is a regular set and $\psi(F(R))$ is a semilinear set. So we have reduced in linear time the membership problem for CUT(R) to a membership problem in the semilinear set $\psi(F(R))$. Since, for a fixed semilinear set, membership can be decided in linear time, the theorem follows. □

2. Implications for graph grammars

In this section we demonstrate how the ideas from Section 1 can be applied to some graph grammar models with fCR. The constructions involved are rather simple – hence, we introduce the necessary notions in an informal way to avoid unnecessary technicalities. All graph grammars we consider generate only languages in NP. This has been explicitly proved only for boundary NLC grammars, see Rozenberg & Welzl (1984). However, also for the two other models this can be easily proved by using standard techniques – hence, we omit further details.

We start with context-free edge replacement grammars (see, e.g. Kreowski, 1979) which constitute a special case of the algebraic approach to graph grammars (see, Ehrig, 1979).

Context-free edge replacement graph grammars (ER grammars)

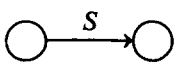
The graphs considered are directed edge labelled graphs. An ER production is a tuple (A, Y, s, t) where A is a label, Y is a graph, and s and t are nodes of Y.

An ER production (A, Y, s, t) is applied to an edge e (of a graph X) labelled by A (see Fig. 2) as follows:

(i) remove e from X;

(ii) add Y to the remainder of X by identifying s with the source node of e and t with the target node of e.

An ER grammar is now a system $G=(N, \Sigma, P, S)$, where N is a finite set of nonterminal labels, Σ is a finite set of terminal labels, P is a finite set of ER productions (A, Y, s, t), with $A\in N$, and Y a graph with labels from $N\cup\Sigma$, and S is a label in N. The graph language $L(G)$ generated by G is the set of all graphs with labels from Σ only which can be derived from the graph



(by applying productions in P as described above). □

The reader might now easily recognize, how we can construct an ER grammar G with NP-complete $L(G)$, following the lines of Section 1.

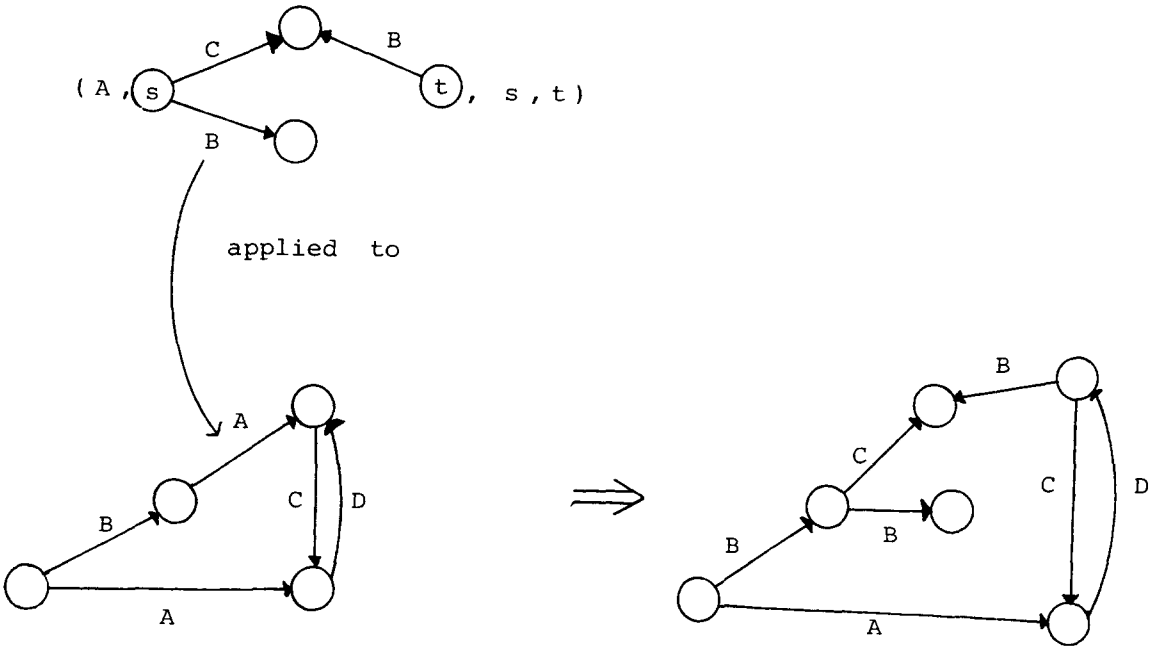
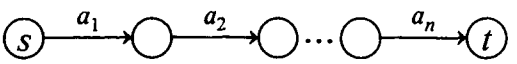


Fig. 2. Application of an ER production.

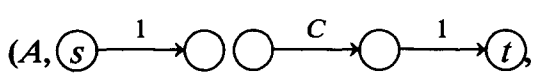
Theorem 2.1. *There is an ER grammar G such that $L(G)$ is an NP-complete language of graphs of maximal degree at most 2.*

Proof. Let $G_0=(N_0, \Sigma_0, P_0, S_0)$ be the linear context-free string grammar from Lemma 1.1. For a word $w=a_1a_2\cdots a_n$, $n\geq 1$, $a_i\in N_0\cup\Sigma_0$, $1\leq i\leq n$, let egra(w) be the graph which can be obtained from



by erasing all edges labelled by ¢. Moreover, let s(egra(w))$=s$ and t(egra(w))$=t$ as indicated above. Consider now the ER grammar $G=(N, \Sigma, P, S)$, where $N=N_0$, $\Sigma=\Sigma_0\setminus\{¢\}$, $P=\{(E, Y, s, t)\,|\,E\to w\in P_0$, such that $Y=$egra(w), $s=s$(egra(w)) and $t=t$(egra(w))$\}$, and $S=S_0$.

For example, the production $A\to 1¢C1$ gives rise to the production



in P_0.

For a multiset M of words in $(\Sigma_0\setminus\{¢\})^*$, let egra($M$) be the disjoint union of all egra(w) with w in M. Then it is easily seen that $M\in$CUT(K), (where $K=L(G_0)$) if and only if egra(M)$\in L(G)$. This shows that $L(G)$ is an NP-complete graph language. □

The next graph grammar type we consider is that of context-free node replacement grammars as they were introduced by Pratt (1971).

Context-free node replacement graph grammars (NR grammars)

The graphs considered are directed node labelled graphs. (For the sake of simplicity we omit the edge labels which are also included by Pratt.) An NR production is a tuple (A, Y, I, O), where A is a label, Y is a graph, and I and O are nodes of Y.

An NR production (A, Y, I, O) is applied to a node v (of a graph X) labelled by A as follows:

(i) remove v from X;

(ii) add Y (disjointly) to the remainder of X;

(iii) edges originally ingoing to v become ingoing edges of I and edges originally outgoing from v become outgoing edges of O.

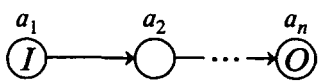
An NR grammar is now a system $G=(N, \Sigma, P, S)$, where N is a finite set of nonterminal labels, Σ is a finite set of terminal labels, P is a finite set of NR productions (A, Y, I, O), with $A \in N$ and Y a graph with labels from $N \cup \Sigma$, and S is a label in N. The graph language $L(G)$ generated by G is the set of all graphs with lables from Σ only which can be derived from the graph

(by applying productions in P as described above). □

Theorem 2.2. *There is an NR grammar G such that $L(G)$ is an NP-complete language of graphs of maximal degree at most 2.*

Proof. The basic idea used here is similar to the one of the proof of Theorem 2.1.

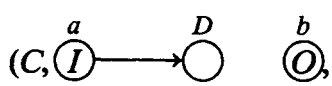
Let $G_0=(N_0, \Sigma_0, P_0, S_0)$ be the string grammar from Lemma 1.1. For a word $w=a_1a_2\cdots a_n$, $n \geq 1$, $a_i \in N_0 \cup \Sigma_0$, $1 \leq i \leq n$, $a_1 \neq ¢$, $a_n \neq ¢$, let ngra(w) be the graph which can be obtained from



by erasing all nodes (with incident edges) which are labelled by ¢. Moreover, let $I(\text{ngra}(w))=I$ and $O(\text{ngra}(w))=O$ as indicated above. (Note that I and O have not been erased, because we assumed $a_1 \neq ¢$ and $a_n \neq ¢$.)

Consider now the NR grammar $G=(N, \Sigma, P, S)$, where $N=N_0$, $\Sigma=\Sigma_0 \setminus \{¢\}$, $P=\{(E, Y, I, O) \mid E \to w \in P_0$, such that $Y=\text{ngra}(w)$, $I=I(\text{ngra}(w))$ and $O=O(\text{ngra}(w))\}$, and $S=S_0$.

For example, the production $C \to aD¢b$ gives rise to the production



in P_0.

Observe that for all productions $E \to w$ in P_0, the first and last symbol of w is not equal to ¢. It is now obvious, how the NP-completeness of CUT($L(G_0)$) implies that of $L(G)$. □

The definition of context-free graph grammars by Della Vigna & Ghezzi (1978) is identical with Pratt's, except for the fact that they require the graph Y in a production (A, Y, I, O) to be connected. The above construction fails for this definition!

We conclude this section with considering boundary node label controlled graph grammars, a subclass of node label controlled (NLC) graph grammars as they were introduced by Janssens & Rozenberg (1980a).

Boundary node lable controlled graph grammars (BNLC grammars)

The graphs considered are undirected node labelled graphs. An NLC production is simply a pair (A, Y) where A is a label and Y is a graph. Let Γ be a set of labels, let (A, Y) be an NLC production with $A \in \Gamma$ and Y a graph with labels from Γ, and let conn be a function from Γ to 2^{Γ}. Let X be a graph with labels from Γ and let v be a node of X. Then the production (A, Y) is applied to v following conn by performing the three steps below:

(i) remove v from X;

(ii) add Y (disjointly) to the remainder of X;

(iii) whenever a (former) neighbour v_1 of v is labelled by b and a node v_2 of Y is labelled by c, and $b \in \text{conn}(c)$, then insert an edge between v_1 and v_2.

A BNLC grammar is a system $G = (N, \Sigma, P, \text{conn}, S)$, where N is a finite set of nonterminal labels, Σ is a finite set of terminal labels, for $\Gamma = N \cup \Sigma$, conn is a function from Γ to 2^{Γ}, and S is a label in N. P is a finite set of NLC productions (A, Y) where (i) $A \in N$ and (ii) Y is a graph with labels from $N \cup \Sigma$, such that no two nodes in Y with labels from N are adjacent. The graph language $L(G)$ generated by G is the set of all graphs with labels from Σ only which can be derived from the graph

by applying productions in P following conn as described above. □

In order to apply our result to Boundary NLC grammars we have to modify the language K from Section 1. Let g be the homomorphism from $\{1', 0'\}^*$ to $\{1, 0\}^*$, defined by: $g(1') = 1$ and $g(0') = 0$. Then

$$K' = \{a^{i_1} w_1 \¢ a^{i_2} w_2 \¢ a^{i_3} \cdots w_{n-1} \¢ a^{i_n} w_n \¢ b^{i_n} u_n \¢ \cdots b^{i_2} u_2 \¢ b^{i_1} u_1 \mid$$
$$n \geq 1;\ i_j \geq 1,\ w_j \in \{0,1\}^+,\ u_j \in \{0', 1'\}^+, \text{ for all } j,\ 1 \leq j \leq n, \text{ and}$$
$$w_j \text{ is word-adjacent to the mirror image of } g(u_{j+1}) \text{ for all } j,$$
$$1 \leq j \leq n-1\}.$$

Then the following result can be easily obtained along the lines of Section 1.

Lemma 2.3. (i) *K' is a linear context-free string language.*

(ii) *The membership problem for* CUT(K') *is NP-complete.*

Proof. We only give the linear context-free grammar which generates K': consider $G_0' = (N_0', \Sigma_0', P_0', S_0)$ where $N_0' = N_0$ (N_0 refers to G_0 in the proof of Lemma 1.1), $\Sigma_0' = \{a, b, 0, 1, 0', 1', \not\in\}$, and P_0' consists of the following productions:

$$S_0 \to S_0 0' \mid S_0 1' \mid C0' \mid C1';$$

$$C \to aCb \mid aA \not\in b \mid aD \not\in b;$$

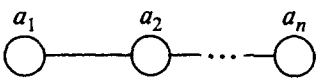
$$A \to 0A1' \mid 1A0' \mid 1B1' \mid 1 \not\in C1';$$

$$B \to iBj \mid i \not\in Cj, \quad i \in \{0,1\},\ j \in \{0',1'\};$$

$$D \to 0D \mid 1D \mid 0 \mid 1. \qquad \square$$

Theorem 2.4. *There is a BNLC grammar G such, that $L(G)$ is an NP-complete language of graphs of maximal degree at most 2.*

Proof. Let $G_0' = (N_0', \Sigma_0', P_0', S_0)$ be the string grammar from Lemma 2.3. For a word $w = a_1 a_2 \cdots a_n$, $n \geq 1$, $a_i \in N_0 \cup \Sigma_0$, $1 \leq i \leq n$, let bgra(w) be the graph which can be obtained from



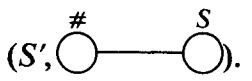
by erasing all nodes (with incident edges) which are labelled by $\not\in$.

Consider now the BNLC grammar $G = (N, \Sigma, P, \text{conn}, S)$ where $N = N_0'$, $\Sigma = \Sigma_0' \setminus \{\not\in\}$, $P = \{(E, \text{bgra}(w)) \mid E \to w \in P_0'\}$, $S = S_0$ and conn is defined by: $\text{conn}(E) = \emptyset$ (the empty set) for $E \in N$; $\text{conn}(d) = \{0, 1, a\}$, for $d \in \{0, 1, a\}$; $\text{conn}(d') = \{0', 1', b\}$, for $d' \in \{0', 1', b\}$. It is now easily seen that $L(G) = \{\text{bgra}(w) \mid w \in L(G_0')\}$ and that the NP-completeness of CUT($L(G_0')$) implies that $L(G)$ is NP-complete. $\square$

In the case of BNLC grammars we can extend the result to the existence of an NP-complete BNLC language (i.e., a language generated by a BNLC grammar) of connected graphs. Note, however, that if the graphs of a BNLC language are both connected and of maximal degree at most k (where k is a fixed integer), then this BNLC language has polynomial time membership (see Rozenberg & Welzl, 1984). Hence, the following theorem together with Theorem 2.4 points at the 'boundary' between polynomial time and NP-complete membership for BNLC languages.

Theorem 2.5. *There is a BNLC grammar G such that $L(G)$ is an NP-complete language of connected graphs.*

Proof. We modify the grammar G from the proof of Theorem 2.4 as follows: introduce a new nonterminal label S' which is the new start label; introduce a new terminal lable # which is added to every set conn(d), $d \in N \cup \Sigma$; $\text{conn}(\#) = \text{conn}(S') = \emptyset$; finally we add the production



The resulting grammar generates now exactly all graphs from $L(G)$ enlarged by a node labelled by # which is adjacent to all other nodes. Clearly, this language contains only connected graphs and it is NP-complete, as $L(G)$ is NP-complete. □

Concerning the fCR, we remark that because of the context-freeness of the observed grammars – the left hand side of a production is a single node or single edge – the fCR reduces to guarantee that if we pick in a graph two nonterminally labelled nodes (or edges) and choose two rules to be applied to these nodes, then the result has to be independent of the order of which node we rewrite first. In the case of ER and NR grammars this is enforced by the embedding mechanism and in the case of BNLC grammars by the fact that there are no adjacent nonterminal nodes in any derived graph.

3. Discussion

In this paper we have shown that if context-free string grammars are equipped with the power of disconnecting, then they can generate NP-complete languages. Thus this 'power of disconnecting' appears to be a basic root of the difficulty in graph grammar parsing as demonstrated in Section 2.

The reader might want to verify that the language K we considered in Section 1 gives not only rise to an NP-complete CUT(K) – rather also the following languages obtained from K are NP-complete:

(1) For a word $w \in \{0, 1, ab, ¢\}^*$ let set(w) be cut(w) interpreted as a set; so set($01¢¢10¢01¢000¢$) = $\{01, 10, 000\}$. Then the language SET(K) = $\{\text{set}(w) \mid w \in K\}$ is NP-complete. The reason why we have chosen cut(w) instead of set(w) (i.e., a multiset interpretation instead of a set interpretation) is given by our interest in graphs: in a graph two isomorphic connected components are considered as different objects.

(2) Consider the language

$$\text{PERMUT}_{¢}(K) = \{w_1 ¢ w_2 ¢ \cdots ¢ w_n \mid \text{there is a permutation } (i_1, i_2, \ldots, i_n) \text{ of } (1, 2, \ldots, n) \text{ such that } w_{i_1} ¢ w_{i_2} ¢ \cdots ¢ w_{i_n} \in K\}.$$

Then PERMUT$_{¢}$(K) is NP-complete which can be seen directly from the NP-completeness of CUT(K).

(3) Consider the language

$$\text{PERMUT}(K) = \{w_1 \$ w_2 \$ \cdots \$ w_n \mid \text{there is a permutation } (i_1, i_2, \ldots, i_n) \text{ of } (1, 2, \ldots, n) \text{ such that } w_{i_1} w_{i_2} \cdots w_{i_n} \in K\},$$

where \$ is a new symbol. Then PERMUT(K) is NP-complete. Note, however, that the language

$$\{w_1 w_2 \cdots w_n \mid \text{there is a permutation } (i_1, i_2, \ldots, i_n) \text{ of } (1, 2, \ldots, n)$$
$$\text{such that } w_{i_1} w_{i_2} w_{i_3} \cdots w_{i_n} \in K\}$$

has an easy membership problem, because a word is in this language if and only if its Parikh vector is an element of the Parikh image of K.

In addition to Theorem 1.7 it should be noted that for a regular set R the membership problem of CUT(R) is solvable in deterministic logarithmic space. In this case it is important to encode multisets in the way indicated in Section 1, as sequences of words rather than as mappings with a finite support. Nevertheless, Theorem 1.4 and Theorem 1.7 are not affected by this question.

References

F.-J. Brandenburg (1982), The computational complexity of certain graph grammars, Lecture Notes in Computer Science 145, 91–99.

F.-J. Brandenburg (1983), On the complexity of the membership problem for graph grammars, M. Nagl and J. Perl, eds., Proceedings of the WG'83 (Universitätsverlag Trauner, Linz) 40–49.

W. Bucher and H. Maurer (1984), Theoretische Grundlagen der Programmiersprachen – Automaten und Sprachen (BI, Mannheim), Reihe Informatik 41.

P. Della Vigna and C. Ghezzi (1978), Context-free graph grammars, Information and Control 37, 207–233.

H. Ehrig (1979), Introduction to the algebraic theory of graph grammars, Lecture Notes in Computer Science 73, 1–69.

R. Frank (1978), A class of linearly parsable graph grammars, Acta Informatica 10, 175–201.

M.R. Garey and D.S. Johnson (1979), Computers and Intractability – A Guide to the Theory of NP-Completeness (Freeman, San Francisco).

D. Janssens and G. Rozenberg (1980a), On the structure of node label controlled graph languages, Inform. Sci. 20, 191–216.

D. Janssens and G. Rozenberg (1980b), Restrictions, extensions, and variations of NLC grammars, Inform. Sci. 20, 217–244.

M. Kaul (1983), Parsing of graphs in linear time, Lecture Notes in Computer Science 153, 206–218.

H.-J. Kreowski (1979), A pumping lemma for context-free graph languages, Lecture Notes in Computer Science 73, 270–283.

T.W. Pratt (1971), Pair grammars, graph languages and string-to-graph translations, J. Comput. System Sci. 5, 560–595.

G. Rozenberg and A. Salomaa (1980), The Mathematical Theory of L Systems (Academic Press, New York).

G. Rozenberg and E.O.R. Welzl (1986), Boundary NLC graph grammars – basic definitions, normal forms, and complexity, Information and Control 69, 136–167.

K. Ruohonen (1975a), Developmental systems with interaction and fragmentation, Information and Control 28, 91–112.

K. Ruohonen (1975b), JL systems with non-fragmented axioms: the hierarchy, Internat. J. Comput. Math. 5, 143–156.

A. Salomaa (1973), Formal Languages (Academic Press, London).

A.O. Slisenko (1982), Context-free grammars as a tool for describing polynomial-time subclasses of hard problems, Inform. Process. Lett. 14, 52–56.

Gy. Turán (1983), On the complexity of graph grammars, Acta Cybernet. 6, 271–281.